SYSTEM MONTORING: LESSONS LEARNED FROM NGOP

Marc Mengel,
Tanya Levshina,
Jim Fromm

Overview

- Why Worry About Monitoring?
- What Worked in NGOp
- What Did NOT Work So Well
- Multiple Views
- Current Architecture
- Propsed Architecture

Why Worry About Monitoring?

Monitoring:

- needs to scale ever larger,
- particularly for GRID

Why Worry About Monitoring?

Most current solutions do not scale well:

Package	**scale**
Ganglia	512 systems
Big Brother	75 systems
Nagios (NetSaint)	650 systems
NGOp	2000 systems

Why Worry About Monitoring?

Grid monitoring needs to scale into the 10^5 systems range.

So what do we learn from scaling to 2000 systems that can help us scale to 200,000?

What Worked with NGOP

- Found outages and trouble predictors promptly, improving system uptimes and availability
- Detected easily missed conditions (1 of 2 cpu's offline, etc.)
- Administrator updated known status prevents alarms for planned outages, and provides notice to users
- Lightweight UDP protocol allows thousands of agents to report to one server

What Worked with NGOP

- Multiple "status engines" providing customized views of hierarchy and error levels for different categories of users
- Python prototyping, with C/C++ where needed for performance
- Nested XML-based notation for configuration
- Database logging of alarms, etc. by asynchronous process, rather than a database at the center

What Didn’t Work so well...

Multiple threads for protocol handling

- needed dual CPUs on central servers to avoid packet loss
- agents didn’t always start all threads, but look like they’re running
- a state-machine, single thread implementation would have been better

What Didn’t Work so well...

Configuration data too amorphous

- editors/web-forms for configuration changes came too late
- full dynamic updates of configuration never got supported
- no standardized naming of elements, clusters, etc.

What Didn’t Work so well...

Naming conventions

- dot-separated tuples looked good at first, but couldn’t talk about off-site/cross-domain entities
- cluster.system.host.component tuples were confusing, often having cluster == host, etc.

Multiple Views

NGOp-style multiple views of monitored systems will be even more important in a Grid universe:

Grid users

need to be able to monitor the systems on which their jobs are running -- even as the job moves to different systems.

Grid operations folks

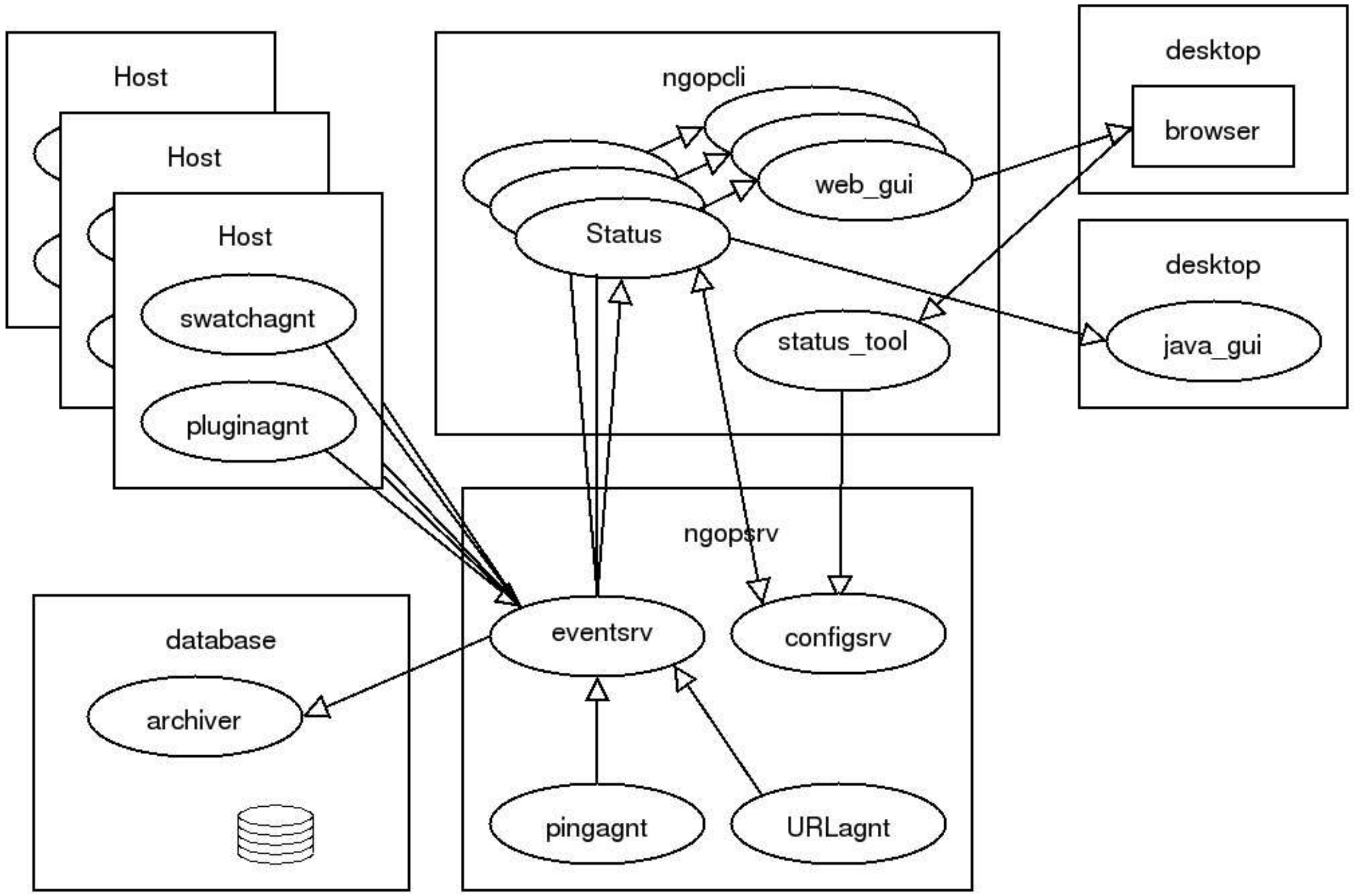
need to be able to:

- spot failures
- find out about planned maintenance and do this accross multiple sites, so they can adjust configurations for distributing jobs, etc.

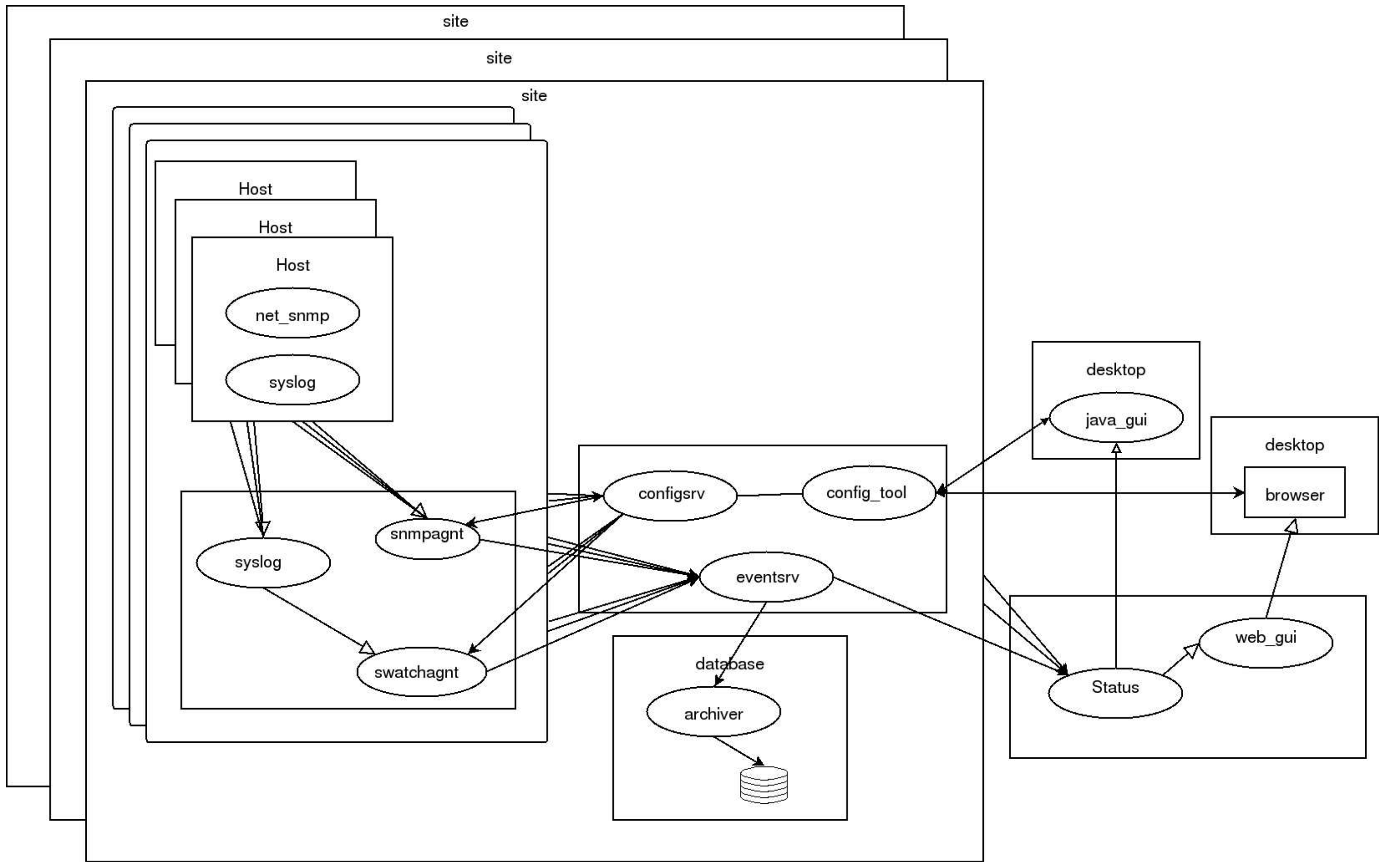
Multiple Views

- Local systems administrators
  - need views of the systems they manage, without the distraction of sites they don’t manage, and with issues that grid users ignore highlighted so they can take preventive measures
- Site operations folks
  - views that show problems actively preventing work getting done, or that might require manual intervention, without trouble predictors that should be left to sysadmins.

Current Archetecture



Proposed Archetecture



Proposed Archetecture

- Each "agency" could, with snmp and swatch (logfile watcher) agents, watch around 200 nodes
- Each site event server should handle up to 5000 agencies.
- Config service should be structured, deal well with multi-layered clusters of similar nodes with exceptions

Proposed Archetecture

- Unlike NGOp, agents should get info on what to watch from config service.
- Community servers could watch multiple sites, and only get info on things at that site they care about
- Could still have special purpose agents on specific nodes (i.e. webserver log watcher, etc.)
- One could run SNMP based tools like MRTG or Cricket on agency nodes, as well.

Conclusions

- NGOp scales, pretty well, but not well enough
- We’ve presented a design that we believe can work.
- If anyone is planning such a project...